UNIVERSIDAD POLITECNICA DE VALENCIA

Primitivas gráficas: dibujando con OpenCV

Apellidos, nombre	Agustí Melchor, Manuel[1] (magusti@disca.upv.es)
Departamento	[1]Dpto. De Ing. De Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSIDAD POLITECNICA DE VALENCIA

1 Resumen

En este artículo vamos a presentar las funciones de dibujo nativas de OpenCV [1-3] que permiten dibujar las figuras geométricas simples, esto es, desde cómo se modifican con una sola operación un conjunto de píxeles de una imagen que tiene un sentido gráfico. A partir de estas se pueden generar figura 2D variadas cambiando los valores de los parámetros.

2 Introducción

Este apartado muestra que se pueden crear contenidos visuales a partir de primitivas vectoriales que son llevadas a cabo sobre las imágenes en formato de mapa de bits dentro de OpenCV. Estas operaciones sirven desde para inicializar el contenido de la imagen, hasta para mostrar resultados al mismo (desde mostrar texto hasta remarcar una zona de la imagen).

Hay que tener en cuenta que las coordenadas de dibujo tienen como origen de coordenadas (0,0) en la parte superior izquierda y toman valores crecientes en el sentido de los ejes, hasta alcanzar el máximo (ancho-1,alto-1) en la esquina inferior derecha. En la Figura 1 se puede ver un instante de la ejecución del ejemplo que comentamos a lo largo de este artículo.

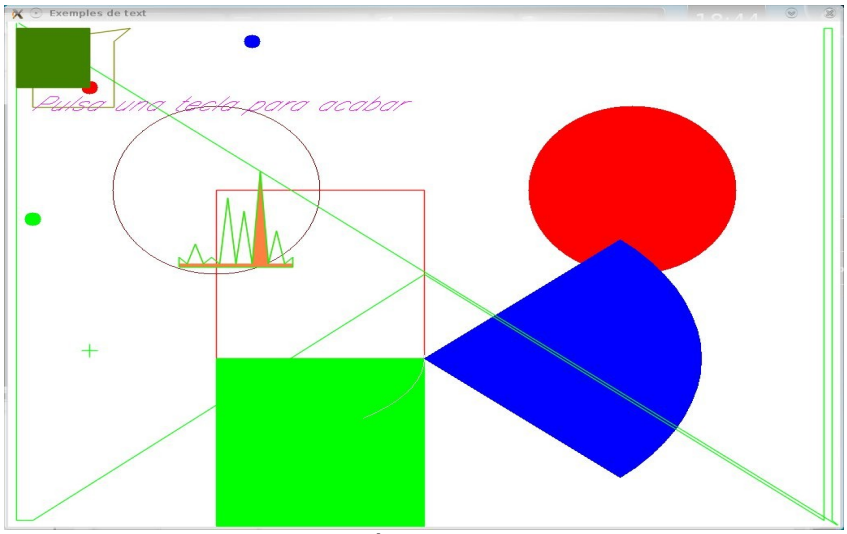

Figura 1: Primitivas básicas de dibujo en OpenCV.

3 Objetivos

Una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Instanciar las primitivas básicas de dibujo que ofrece OpenCV como:rectángulos, círculos, polilíneas y texto.
- Combinarlas para obtener elementos gráficos de uso común en aplicaciones de procesado de imágenes.

4 Desarrollo

Las funciones que abordamos nos permiten, como se muestra en la Figura 1, ir pintando aleatoriamente diferentes primitivas con diferentes parámetros. Incluso, trozos de un mapa de bits que se hubiera cargado previamente. Esto se realiza a base a ejecutar las diferentes instrucciones sobre una misma imagen, el mismo mapa de bits (la misma imagen) se visualiza de manera periódica para que se puedan ver las acciones realizadas sobre ella.

Hay que prestar atención a qué índice se refiera a filas o a columnas de una coordenada de dibujo y su correspondencia en la imagen.

```
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void rectangles( IplImage *image, char * nomFinestra );
void circles( IplImage *image, char *nomFinestra );
void polilinies( IplImage *image, char *nomFinestra );
void printfXY( IplImage* image, char *nomFinestra,  char *mensatge );

int main( int argc, char* argv[] )
{
  char nomFinestraPpal[1024] = "Exemples de text";
  int ampleFinestra = 1024, altFinestra = 768;
  int nVoltes = 5, i, j, tecla = 0;
  IplImage* image;
  char mensatge[1024];

  image = cvCreateImage(cvSize(ampleFinestra, altFinestra), IPL_DEPTH_8U,  3);
  cvSet(image, CV_RGB(255,255,255), NULL );
  printf("Sobre una finestra de columnas x filas: %dx%d, se dispone:\n",
          ampleFinestra, altFinestra);
  cvNamedWindow(nomFinestraPpal, CV_WINDOW_AUTOSIZE);
  cvShowImage(nomFinestraPpal, image);

  rectangles( image, nomFinestraPpal );
  circles( image, nomFinestraPpal );
  polilinies( image, nomFinestraPpal );
  printfXY(image, nomFinestraPpal, "Pulsa ESC per acabar" );

  tecla = cvWaitKey(0) & 255;
  cvReleaseImage(&image);
  cvDestroyWindow(nomFinestraPpal);

  return 0;
} // Fi de "main( ..."
```

Listado 1. Ejemplos de primitivas gráficas en OpenCV.

4.1 Programa principal

En primer lugar mostraremos el programa principal e iremos desglosando en los diferentes apartados, diferentes primitivas gráficas.

El esquema de trabajo utilizado se basa en el siguiente esquema:

- Inicializar una imagen.
- Pintar sobre ella
  - rectángulos: `cvRectangle`.
  - círculos: `cvCircle` y `cvEllipse`.
  - polilíneas: `cvPolyLine`.
  - `polilíneas: cvPolyLine`.
  - `texto: cvPutText`.
- Mostrar la imagen y esperar una tecla.
- Liberar recursos y terminar.

Este código se limita a crear la ventana, la imagen e inicializarla a color blanco con el uso de la macro *CV_RGB*, en lugar de *cvScalar* . Las llamadas a las funciones que "pintan" los elementos gráficos son las encargadas de decidir qué píxeles de la imagen tomarán nuevos valores de manera acorde con la definición de estás figuras básicas, evitando tener que calcularlos uno mismo.

```
void rectangles( IplImage *image, char * nomFinestra )
{
 int i, j;
 int altFinestra = image->height,
     ampleFinestra = image->width, fila, columna,
    thickness = 1, line_type = CV_AA, shift = 0;

 fila = (int)( image->height / 3 );
 columna = (int)( image->width / 4 );

 cvRectangle( image, cvPoint(columna, fila), cvPoint(2*columna, 2*fila),
              CV_RGB(255,0,0), thickness, line_type, shift);
 printf("Rectángulo en (col, fila) %d,%d a %d,%d\n", columna, fila, 2*columna, 2*fila);
 cvRectangle( image, cvPoint(columna, 2*fila), cvPoint(2*columna, 3*fila),
              CV_RGB(0, 255,0), CV_FILLED, line_type, shift);
 printf("Rectángulo relleno en (col) %dx%d..%dx%d\n", columna, 2*fila, 2*columna, 3*fila );

}// Fi de "void rectangles( ..."
```

Listado 2. Ejemplos de rectángulos en OpenCV.

4.2 Rectángulos

La función *cvRectangle* permite dibujar un rectángulo sobre una imagen ya creada. Hay que indicar el área del mismo, como las coordenadas de la esquina superior izquierda y las de la inferior derecha. También se ha de especificar el tipo de línea circundante, lo que permite indicar si rellena el interior del mismo

o no. El código del listado 2 muestra cómo visualizar un rectángulo vacío y uno relleno.

El lector deberá observar que se dan algunos valores fijos, mientras que otros se obtienen a partir de las propiedades de la imagen donde se va a dibujar, por lo tanto variarán si lo hace el tamaño de la imagen. De esta forma se diferencia entre valores de coordenadas y propiedades de visualización del objeto.

```
void circles( IplImage *image, char *nomFinestra )
{
 int altFinestra = image->height,
    ampleFinestra = image->width, fila, columna, radio;
 CvScalar color;
 int thickness = 1, line_type = 8, shift = 0;
 CvSize axes;
 double angle = 0, start_angle = 45, end_angle = -45;

 cvCircle( image, cvPoint(100, 100), 10, CV_RGB(255,0,0),
         CV_FILLED, line_type, shift );
 printf("Circunferencia en (fila, col) %dx%d\n", 100, 10);
 cvCircle( image, cvPoint(30, 300), 10, CV_RGB(0,255,0),
         CV_FILLED, line_type, shift );
 printf("Circunferencia en (fila, col) %dx%d\n", 300, 10);
 cvCircle( image, cvPoint(300, 30), 10, CV_RGB(0,0,255),
         CV_FILLED, line_type, shift );
 printf("Circunferencia en (fila, col) %dx%d\n", 300, 10);

 fila = (int)( image->height / 3 );
 columna = (int)( image->width / 4 );
 radio = MIN( image->width/6, image->height/6 );

 cvCircle( image, cvPoint(columna, fila), radio, CV_RGB(128,0,0),
         thickness, line_type, shift );
 printf("Circunferencia en (fila, col) %dx%d\n", fila, columna);
 cvCircle( image, cvPoint(3*columna, fila), radio, CV_RGB(255,0,0),
         CV_FILLED, line_type, shift );
 printf("Círculo relleno en (fila, col) %dx%d\n", 2*fila, 3*columna);

 fila = 2 * (int)( image->height / 3 );
 axes = cvSize( 2*radio, radio);
 cvEllipse( image, cvPoint(columna, fila), axes, angle, start_angle, end_angle,
          CV_RGB(255,255,255), thickness, line_type, shift );
 printf("Elípse en (fila, col) %dx%d\n", fila, columna);
 axes = cvSize( image->width/3, image->height/3);
 cvEllipse( image, cvPoint(2*columna, fila), axes, angle, start_angle, end_angle,
          CV_RGB(0,0,255), CV_FILLED, line_type, shift );
 printf("óvalo en (fila, col) %dx%d\n", fila, 2*columna);


 cvShowImage(nomFinestra, image);
}// Fi de "void circles(..."
```

Listado 3. Ejemplos de círculos y elipses en OpenCV.

4.3 Círculos

La función *cvCircle* permite especificar un círculo o una circunferencia. Para ello se especifica la posición del centro del mismo, el radio y las propiedades de la línea exterior y el relleno. Familiarícese con las instrucciones del listado Error: No se encuentra la fuente de referencia para identificar estos elementos.

Análogamente, *cvEllipse* permite dibujar una ... ¡Acertaste, una elipse! A partir del punto central de esta, la longitud de los ejes mayor y menor y la abertura de arco, lo que permite dibujar desde un óvalos hasta semicírculos y arcos.

4.4 Líneas y polilíneas

Aunque hemos comenzado esta excursión por los rectángulos podríamos haber empezado dibujando un punto, que sería equivalente a cambiar un píxel de una imagen, por lo que no existe una función específica para ello. Ya tiene dificultad calcular los puntos de una recta y por ello existe la función *cvLine*.

Esta ya no es compleja a la vista de las anteriores, pero para que se vea cómo usarla se propone en el listado 4 una función de dibuja una cruz en la posición que se indica como parámetro y con un ancho dados.

```
void  pintaCruz( IplImage* image, int fila, int columna, int ancho )
{
  int thickness = 1,
      line_type = CV_AA,
      shift = 0;

  cvLine(image, cvPoint(columna-ancho,fila), cvPoint(columna+ancho, fila), cvScalar(0,255,0,
1), thickness, line_type, shift );
  cvLine(image, cvPoint(columna, fila-ancho), cvPoint(columna, fila+ancho), cvScalar(0,255,0,
1), thickness, line_type, shift );

}// Fi de "void  pintaCruz( ..."
```

Listado 4. Ejemplos de uso de líneas, para dibujar una cruz, en OpenCV.

¿Fácil, verdad? Se me ocurre proponer ahora algo más interesante pero lo dejo para el fina: un reloj analógico.

Es interesante ver que se pueden dibujar gráficas complejas, el listado completo al que se hace referencia en este apartado es el 5, con la versión poligonal: *cvPolyLine*. Esta necesita al menos lista de puntos, indicarle si ha de cerrar automáticamente el polígono y la definición del tipo de línea o relleno de la figura.

Las versiones “con relleno” de los polígonos se crean con *cvFillConvexPoly* o *cvFillPoly*. Se ha incluido[1] un ejemplo complejo que el lector habrá de identificar en la ejecución del código que estamos exponiendo. Si es necesario, se recomienda modificar el código para pararlo después de cada figura dibujar y mostrar mayor número de mensajes en pantalla.

---

[1]Extraido de <http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html>: "Introduction to programming with OpenCV" de Gady Agam; *Department of Computer Science; January 27, 2006; Illinois Institute of Technology*.

```
void polilinies( IplImage *image, char * nomFinestra )
{
 int altFinestra = image->height,
   ampleFinestra = image->width,
   fila, columna, nPunts;
 CvPoint *puntos=0;
 int thickness = 1, line_type = CV_AA, shift = 0;

 fila = image->height;
 columna = image->width;
 nPunts = 10;

 puntos=(CvPoint*)malloc( nPunts * sizeof(puntos[0]));

 // Punts en format (columna, fila)
 puntos[0] = cvPoint(0, 0);
 puntos[1] = cvPoint(columna/2, 0);
 puntos[2] = cvPoint(columna/2, 10);
 puntos[3] = cvPoint(0, 10);
 puntos[4] = cvPoint(0, 20);
 puntos[5] = cvPoint(columna/2, 20);
 puntos[6] = cvPoint(columna/2, 30);
 puntos[7] = cvPoint(0, 30);
 puntos[8] = cvPoint(0, 40);
 puntos[9] = cvPoint(columna/2, 40);
 cvPolyLine( image, &puntos, &nPunts, 1, 0,
              CV_RGB(255,0,0), thickness, 5*line_type, shift+10);
 printf("Polilínea\n" );

 puntos[0] = cvPoint(10, 0);
 puntos[1] = cvPoint(10, fila-10);
 puntos[2] = cvPoint(20, fila-10);
 puntos[3] = cvPoint(30, fila-10);
 puntos[4] = cvPoint(columna/2, fila/2);
 puntos[5] = cvPoint(columna-20, fila-10);
 puntos[6] = cvPoint(columna-20, 10);
 puntos[7] = cvPoint(columna-10, 10);
 puntos[8] = cvPoint(columna-10, fila-10);
 puntos[9] = cvPoint(columna, fila);
 // Si la rellenas, el último punto lo une con el primero, sin preguntar.
 cvPolyLine( image, &puntos, &nPunts, 1, 1,
              CV_RGB(0,255,0), CV_FILLED, line_type, shift);
 printf("Polilínea  rellena \n" );

// "Introduction to programming with OpenCV";   Gady Agam
CvPoint  curve1[]={10,10,  10,100,  100,100,  100,10};
CvPoint  curve2[]={30,30,  30,130,  130,130,  130,30,  150,10};
CvPoint* curveArr[2]={curve1, curve2};
int      nCurvePts[2]={4,5};
int      nCurves=2, isCurveClosed=1, lineWidth=1;
 cvPolyLine(image, curveArr, nCurvePts, nCurves, isCurveClosed,
          cvScalar(0,128,128, 0), lineWidth, line_type, shift);
 printf("Pueden dibujarse más de una a la vez.\n");
//

 printf("Pueden dibujarse polígonos.\n");
```

```
 cvFillConvexPoly(image,curve1,nCurvePts[0], cvScalar(0,128,64, 9), line_type, shift);

 // printf("y varios de ellos.\n");
 //cvFillPoly(image,curveArr,nCurvePts,nCurves,cvScalar(0,128,64, 9), line_type, shift);

 // Una creu
 printf("Creu de 10,100 a nCols,100 x 300,90, 300,110\n");
 pintaCruz( image, 500, 100, 10);

  cvShowImage(nomFinestra, image);
}// Fi de "void pPolilinies( ..."
```

Listado 5. Ejemplos de figuras derivadas de líneas, primitivas gráficas en OpenCV.

```
void printfXY( IplImage* image, char *nomFinestra,  char *mensatge )
{
  double hScale = 1.0,  vScale = 1.0,  lineWidth = 1.0,  italicScale = 1.0;
  int tipoLletra = CV_FONT_HERSHEY_SIMPLEX, columna = 35, fila = 135;
  CvFont font;

  cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX, hScale,vScale, italicScale, lineWidth, 8);
  cvPutText (image,"Pulsa una tecla para acabar", cvPoint(columna, fila ), &font,
CV_RGB(255,0,255));
  printf("Texto en (col, fila) %dx%d\n", columna, fila );
  cvShowImage(nomFinestra, image);

}// Fi de "void printfXY( ..."
```

Listado 6. Ejemplos de primitivas gráficas en OpenCV.

4.5 Texto en modo gráfico

Queda una “figura” un tanto especial que dibujar: el texto. Se puede pensar en los caracteres como figuras con una forma dada (si se considera un tipo de letra en concreto). Se podría hacer con las anteriores instrucciones ... pero créeme. Esta es mucho más conveniente para la salud mental del arquitecto de una aplicación que utiliza visión por computador, hasta para la de un programador gráfico.

El listado 8 muestra los pasos típicos, no hay muchos tipos de letra definidos actualmente en OpenCV, pero todo llegará. Concentrémonos en extraer del ejemplo cómo especificar uno, así como el resto de parámetros que se pueden utilizar al respecto del texto.

Señalar que, a diferencia de una consola de texto, aquí somos nosotros los encargados de manejar el “salto de línea” y otros detalles que son implícitos en un terminal. ¿Quién ha dicho que el texto sea fácil?

5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto ejemplos de uso de las primitivas básicas de dibujo vectorial presentes en OpenCV de forma nativa. El lector ha podido experimentar con.

- Operaciones de dibujo de líneas, rectángulos, círculos y sus derivados.
- Los parámetros que definen tales figuras geométricas.

Ahora toca ponerse a experimentar. Dos propuestas, una ya la he mencionado al hablar de cómo dibujar líneas: ¿se te ocurre cómo realizar un reloj como el de la figura 2? Lo difícil de este ejemplo es la determinación de la posición de los extremos de la "varillas" de las horas, minutos y segundos (en rojo, verde y azul respectivamente).

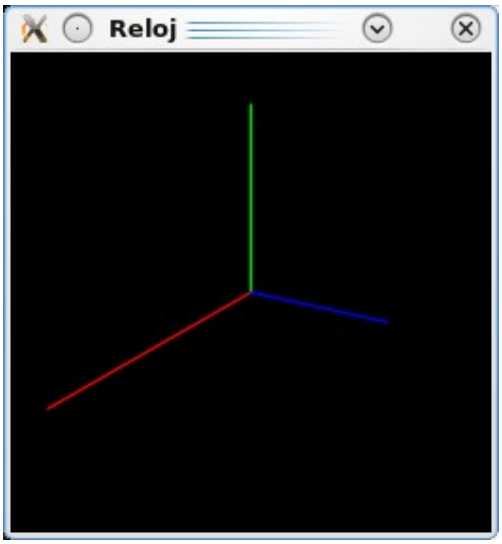

Figura 2: Captura de un instante de la ejecución: las 20:00:17.

Y la otra: ¿qué tal hacer (al estilo de la fig. 3) que los valores de los parámetros de las primitivas se obtengan de forma aleatoria y que copien trozos de una imagen cargada de disco u obtenida de la cámara?